IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | | |
|---|---|---|---|
| Applicant | : Parent et al. | Art Unit | : 2142 |
| Serial No. | : 09/965,117 | Examiner | : Michael D. Meucci |
| Filed | : September 26, 2001 | Conf. No. | : 1762 |
| Title | : MARKED FOREIGN DATA BLOCKS | | |

**Mail Stop Appeal Brief - Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

In accordance with appellant's Notice of Appeal filed May 30, 2006, appellant submits this Appeal Brief.

**(1) Real Party in Interest**

The real party in interest is Adobe Systems Incorporated.

**(2) Related Appeals and Interferences**

None.

**(3) Status of Claims**

Claims 1-18 and 21-28 are pending and stand rejected. Applicant appeals the rejection of claims 1-18 and 21-28.

**(4) Status of Amendments**

There are no unentered amendments.

**(5) Summary of Claimed Subject Matter**

Applicant's specification relates to embedding and extracting data blocks in a host data file or document. *See* Specification, p. 1. A foreign data block is embedded in a host data file in a way that allows it to be found and possibly modified by a foreign application. The foreign application can locate embedded foreign data blocks in host data files even when the host data files have formats that the foreign application does not recognize. *See* Specification, p. 1. A system receives a foreign data block to be embedded into a host file. *See* Specification, p. 5 and

FIG. 1. The characteristics of the foreign data block are determined including computing the total length and determining the type of encoding of the data block. A header is generated for the foreign data block that includes information such as the characteristics of the foreign data block and a unique identifier that is designed to be distinguishable from all other data in the host data file. The header is marked by a byte pattern that identifies the encoding of the foreign data block. *See* Specification, p. 10 and TABLE 3, which is reproduced below.

| Encoding | Byte Pattern |
|---|---|
| 16-bit | 0x3C 0x00 0x3F 0x00 0x78 0x00 0x70 0x00 0x61 0x00 0x63 0x00 0x6B 0x00 0x65 0x00 0x74 0x00 0x20 0x00 0x62 0x00 0x65 0x00 0x67 0x00 0x69 0x00 0x6E 0x00 0x3D (0x00) |
| 8-bit or multiple encoding | 0x3C 0x3F 0x78 0x70 0x61 0x63 0x6B 0x65 0x74 0x20 0x62 0x65 0x67 0x69 0x6E 0x3D |
| 32-bit | 0x3C 0x00 0x00 0x00 0x3F 0x00 0x00 0x00 0x78 0x00 0x00 0x00 0x70 0x00 0x00 0x00 0x61 0x00 0x00 0x00 0x63 0x00 0x00 0x00 0x6B 0x00 0x00 0x00 0x65 0x00 0x00 0x00 0x74 0x00 0x00 0x00 0x20 0x00 0x00 0x00 0x62 0x00 0x00 0x00 0x65 0x00 0x00 0x00 0x67 0x00 0x00 0x00 0x69 0x00 0x00 0x00 0x6E 0x00 0x00 0x00 0x3D |

**(6) Grounds of Rejection**

Claims 1-2, 4, 6-8, 10-13, 23-25 and 27-28 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent (U.S. Pat. No. 5,778,359) in view of Backlund (OOE: A Compound Document Framework).

Claims 14-16 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent in view of Lonnroth (U.S. Pat. No. 6,826,597 B1).

Claims 17-18 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent and Lonnroth in view of Backlund.

Claims 19-22 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Backlund in view of Stent.

Claim 3 stands rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent and Backlund in view of Erickson (U.S. Pat. Pub. No. 2004/0210535) and Parks (U.S. Pat. No. 6,850,228).

Claim 5 stands rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent and Backlund in view of Parks.

Claims 9 and 26 are rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Stent and Backlund in view of Walsh (U.S. Patent No. 6,810,429).

**(7) Argument**

**I. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent in view of Backlund**

Claim 1

Claim 1 stands rejected as unpatentable over Stent and Backlund. Stent discloses a system and method for automatically determining a file record format. *See* Abstract. A typical file is a database file 16, which can include a header 30 that contains information about the contents of the file. After the header begins the main body 32, which comprises one or more file records. Each record includes a number of different fields. Each field contains a predetermined data type. The file 16 ends with a trailer 34. *See* col. 3, lines 49-56.

Stent does not disclose selecting, based on the character encoding format of the foreign data block, a byte pattern that indicates a presence of a header, as required by Claim 1. Stent discloses skipping the file header and analyzing the main body in order to determine a record format. *See* col. 5, lines 18 23. One way to skip the header is to search at the beginning of the file for the word "HEADER" and then skip over the portion of the file between the word "HEADER" and a subsequent carriage return or line feed. *See* col. 5, lines 12 18. However, Stent does not teach that the word "HEADER" was selected based on the character encoding format of the foreign data block, as required by claim 1. Moreover, the word "HEADER" does not specify a byte pattern, which claim 1 requires, because there are different byte patterns that can represent this word. It follows that Stent does not disclose selection of a byte pattern that indicates the presence of a header based on a determined character format.

The relied upon portions of Backlund do not remedy the deficiencies in Stent. Backlund describes a framework where so-called compound documents can contain embedded objects. *See* p. 2. Backlund discloses that an embedded object is stored in a host document as image data, document data, and source data. *See* p. 3, § Basic Storage. Image data contains data used to

render an embedded object. Document data contains internal structures of the embedded object. And source data contains information about which application was used to create the embedded object. Assuming for argument's sake that embedded objects in Backlund are equivalent to foreign data blocks, these portions of Backlund do not teach or suggest generating packing data that describes the characteristics of the foreign data block, including data identifying the beginning and end of the foreign data block. Furthermore, the cited portions of Backlund do not teach or suggest selection of a byte pattern for generated packing data that indicates the presence of a header based on a determined character format, as required by claim 1.

Accordingly, the applicant respectfully submits that claim 1, and claims 2, 4, 6-8, and 10-13, which depend from 1, are allowable.

Claim 21

Claim 21 stands rejected as unpatentable over Stent and Backlund. Claim 21 includes limitations similar to those of claim 1. As addressed above, claim 1 is not obvious in view of Stent and Backlund. For at least this reason, claim 21, and claims 23-25 and 27-28, which depend from claim 21, are in condition for allowance

**II. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent in view of Lonnroth.**

Claim 14

Claim 14 stands rejected as unpatentable over Stent and Lonnroth. Claim 14 recites in part, when the byte pattern is found, determining a character encoding format of the header. Stent does not disclose determining a character encoding format of a header when a byte pattern indicating the presence of the header is found, as required by claim 14. The portions of Stent relied upon by the examiner fail to teach or suggest this feature (col. 5, lines 12-19):

> The next step is to determine whether file 16 contains headers or trailers, step 84. One method of determining if file 16 contains a header is to search the beginning of file 16 for the word "HEADER" immediately after a carriage return, new line or linefeed character. The word "HEADER" will usually be subsequently followed by another carriage return or a linefeed, which indicates the end of the header 30, FIG. 2. The header 30 can then be skipped over.

The relied upon section of Stent above discloses that a file can have a header section which can be demarked with the word "HEADER". There is no mention of determining the character encoding format of the header. In fact, Stent teaches that character encoding is determined before a header is searched for in order to convert the file to ASCII format. *See* FIG 4A and col. 4, lines 20-31.

Lonnroth discloses a system for a mobile phone 210 to communicate with a gateway computer 202 over a network 212. *See* col. 3, line 63 – col. 3, line 6, and FIG. 2. The system allows mobile phones to retrieve data from data sources that do not necessarily support the same protocols and formats as the mobile phones. *See* col. 3, lines 14-19. But the relied upon portions of Lonnroth fail to remedy the deficiencies in Stent.

Accordingly, the applicant respectfully submits that claim 14, and claims 15-16 which depend from 14, are allowable.

**III. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent and Lonnroth in view of Backlund.**

Claims 17-18

Claims 17 and 18 stand rejected as unpatentable over Stent, Lonnroth and Backlund. Claims 17 and 18 depend from claim 14. As addressed above, claim 14 is not obvious in view of Stent and Lonnroth. The relied upon portions of Backlund fail remedy the deficiencies in Stent and Lonnroth. Accordingly, the applicant respectfully submits that claims 17 and 18 are allowable.

**IV. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Backlund in view of Stent.**

Claims 19-22

Claims 19-22 stand rejected as unpatentable over Backlund and Stent. The rejection of claims 19 and 20 is moot, as these claims are canceled. Claims 21 and 22 contain limitations similar to those found in claim 1. As addressed above, claim 1 is not obvious in view of

Backland and Stent. Accordingly, the application respectfully submits that claims 21 and 22 are allowable.

**V. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent and Backlund in view of Erickson and Parks.**

Claim 3

Claim 3 stands rejected as unpatentable over Stent, Backlund, Erickson and Parks. Claim 3 depends from claim 1. As addressed above, claim 1 is not obvious in view of Stent and Backlund. And the relied upon portions of Erickson and Parks fail to remedy the deficiencies in Stent and Backlund. Accordingly, the applicant respectfully submits that claim 3 is allowable.

**VI. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent and Backlund in view of Parks.**

Claim 5

Claim 5 stands rejected as upatentable over Stent, Backlund and Parks. Claim 5 depends from claim 1. As addressed above, claim 1 is not obvious in view of Stent and Backlund. And the relied upon portions of Parks fail to remedy the deficiencies in Stent and Backlund. Accordingly, the applicant respectfully submits that claim 5 is allowable.

**VII. Rejection under 35 U.S.C. § 103(a) as being unpatentable over Stent and Backlund in view of Walsh.**

Claims 9 and 26

Claims 9 and 26 stand rejected as unpatentable over Stent, Backlund and Walsh. Claim 9 depends from claim 1 and claim 26 depends from claim 21. As addressed above, claims 1 and 21 are patentable in view of Stent and Backlund. And the relied upon portions of Walsh fail to remedy the deficiencies in Stent and Backlund. Accordingly, the applicant respectfully submits that claims 9 and 26 are allowable.

The appeal brief fee in the amount of $500 is being paid concurrently herewith on the Electronic Filing System (EFS) by way of Deposit Account authorization. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 7/31/06

Daniel J. Burns
Reg. No. 50,222

**Customer No. 21876**
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

50356993.doc

## Appendix of Claims

1. A method for embedding a foreign data block in a host data file, comprising:

receiving a host data file, the host data file having a host data file format;

receiving a foreign data block;

determining characteristics of the foreign data block, including a character encoding format of the foreign data block;

generating packing data that describes the characteristics of the foreign data block, including data identifying the beginning and end of the foreign data block and further including an identifier designed to be distinguishable from all other data in the host data file, wherein

generating packing data includes selecting, based on the character encoding format of the foreign data block, a byte pattern that indicates a presence of a header, and including the byte pattern in the packing data; and

embedding the packing data and the foreign data block as a foreign data block packet in the host data file.

2. The method of claim 1, wherein:

generating packing data includes generating the header for the foreign data block, the header including the byte pattern and the identifier and indicating the beginning of the foreign data block packet and the beginning of the foreign data block.

3. The method of claim 2, wherein:

generating a header includes generating a header that indicates the end of the foreign data block packet.

4. The method of claim 2, wherein:

generating packing data includes generating a trailer for the foreign data block, the trailer indicating the end of the foreign data block.

5. The method of claim 2, wherein:

generating a header includes generating a header that indicates the end of the foreign data block.

6. The method of claim 1, further comprising:
including padding in the foreign data block packet to allow in place modifications of the foreign data block that cause the foreign data block to expand.

7. The method of claim 6, wherein:
determining characteristics of the foreign data block includes determining a size of the foreign data block; and
the amount of padding is a function of the size of the foreign data block.

8. The method of claim 1, wherein:
the foreign data block is a data block not native to the host file format.

9. The method of claim 1, wherein the foreign data block is an Extensible Markup Language (XML) document and the host file is in a non-XML format.

10. The method of claim 9, wherein:
determining characteristics of the foreign data block includes determining a byte order of the foreign data block; and
generating packing data includes generating information for specifying the byte order and character encoding format of the foreign data block, the character encoding format being one of an 8, 16, or 32 bit Unicode format.

11. The method of claim 1, wherein generating an identifier includes generating a different identifier for each different type of foreign data block when there are multiple types of foreign data blocks in the host data file.

12. The method of claim 1, wherein the foreign data block includes metadata information that describes the host data file.

13. The method of claim 12, wherein:
receiving a host data file includes receiving a host data file having a non XML format.

14. A computer program product, tangibly stored on a machine readable medium, comprising instructions operable to cause a programmable processor to:

receive a host data file; and

search for a header that indicates the beginning of an embedded foreign data block packet that contains a foreign data block, the foreign data block having a format that is recognizable by the computer program, the header including an identifier designed to be distinguishable from all other data in the host data file, the header further describing the characteristics of the foreign data block, wherein searching for the header comprises:

scanning byte by byte for a byte pattern that indicates a presence of a header; and

when the byte pattern is found, determine a character encoding format of the header and scan character by character using the character encoding format to search for the identifier, and, if the identifier is found, process the header or, if an identifier is not found, scan a remaining portion of the host data file byte by byte for the byte pattern.

15. The computer program product of claim 14, further comprising instructions to:
process the foreign data block.

16. The computer program product of claim 15, further comprising instructions to:
stop processing the foreign data block when a trailer is detected, wherein the trailer indicates the end of the foreign data block.

17. The computer program product of claim 16, further comprising instructions to:
modify the foreign data block as specified by a user;
ensure that the modified foreign data block fits in the foreign data block packet; and
re embed the modified foreign data block in place of the original foreign data block.

18. The computer program product of claim 16, further comprising instructions to:

modify the foreign data block as specified by a user;

rewrite the foreign data block packet;

ensure that the re written foreign data block packet is the same size as the original foreign data block packet; and

re embed the re written foreign data block packet in place of the original foreign data block packet.

19-20. (Canceled)

21. A computer program product, tangibly stored on a machine readable medium, for embedding a foreign data block in a host data file, comprising instructions operable to cause a programmable processor to:

receive a host data file, the host data file having a host data file format that is a native file format for a host application;

receive a foreign data block, the foreign data block being a data block that is not native to the host data file format;

determine characteristics of the foreign data block, including a character encoding format of the foreign data block;

generate information that describes the characteristics of the foreign data block, including information identifying the beginning and end of the foreign data block and further including an identifier designed to be distinguishable from all other data in the host data file, wherein generating information identifying the beginning of the foreign data block includes selecting, based on the character encoding format, a byte pattern that indicates a presence of the information marking the beginning of the foreign data block; and

embed the information and the foreign data block as a foreign data block packet in the host data file.

22. A computer program product, tangibly stored on a machine readable medium, for embedding metadata in a host data file having a non XML format, comprising instructions operable to cause a programmable processor to:

receive a host data file having a format that is not XML and that is a native file format for a host application;

receive metadata having a format that is not native to the host data file format;

determine characteristics of the metadata, including a character encoding format of the metadata;

generate information that describes the characteristics of the metadata, including information identifying the beginning and end of the metadata and further including an identifier designed to be distinguishable from all other data in the host data file, wherein generating information identifying the beginning of the metadata includes selecting, based on the character encoding format, a byte pattern that indicates a presence of the information marking the beginning of the metadata; and

embed the information and the metadata as a packet in the host data file.

23. The product of claim 21, further comprising instructions to:
generate a header for the foreign data block, the header including the identifier and indicating the beginning of the foreign data block packet and the beginning of the foreign data block.

24. The product of claim 21, further comprising instructions to:
generate a trailer for the foreign data block, the trailer indicating the end of the foreign data block.

25. The product of claim 21, further comprising instructions to:
include padding in the foreign data block packet to allow in place modifications of the foreign data block that cause the foreign data block to expand.

26. The product of claim 21, wherein the foreign data block is an Extensible Markup Language (XML) document and the host file is in a non-XML format.

27. The product of claim 26, further comprising instructions to:

determine a byte order of the foreign data block; and

generate a header that includes information for specifying the byte order and encoding format of the foreign data block, the encoding format being one of an 8, 16, or 32 bit Unicode format.

28. The product of claim 21, further comprising instructions to:

generate a different identifier for each different type of foreign data block when there are multiple types of foreign data blocks in the host data file.

## Evidence Appendix

NONE.

## Related Proceedings Appendix

NONE.